BIO-INSPIRED NOISE ROBUST AUDITORY FEATURES

A Thesis
Presented to
The Academic Faculty

by

Ailar Javadi

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical Engineering

Georgia Institute of Technology
August 2012

BIO-INSPIRED NOISE ROBUST AUDITORY FEATURES

Approved by:

Dr. David Anderson, Advisor
School of Electrical Engineering
Georgia Institute of Technology

Dr. Mark Clements
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Christopher Rozell
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: June 2012

ACKNOWLEDGEMENTS

I want to thank my advisor Dr. David Anderson for his full support and for providing me with resources and guidance. I also thank my committee members for their support and valuable input.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Nowadays, speech recognition systems are everywhere. The purpose of this work is to investigate a series of biologically inspired modifications to state-of-the-art Mel-frequency cepstral coefficients (MFCCs) that may improve automatic speech recognition results. We have provided recommendations to improve speech recognition results depending on signal-to-noise ratio levels of input signals. This work has been motivated by noise-robust auditory features (NRAF) in [11] . The feature extraction technique used in our analysis is the same as in [11] in which after a signal is filtered using bandpass filters, a spatial derivative step is used to sharpen the results, followed by an envelope detector (rectification and smoothing) and down-sampling for each filter bank before being compressed. DCT is then applied to the results of all filter banks to produce features. The Hidden-Markov Model Toolkit (HTK) [20] is used as the recognition back-end to perform speech recognition given the features we have extracted. In this work, we have investigated the role of filter types, window size, spatial derivative, rectification types, smoothing, down-sampling and compression and compared the final results to state-of-the-art Mel-frequency cepstral coefficients (MFCC). A series of conclusions and insights are provided for each step of the process. The goal of this work has not been to outperform MFCCs; however, we have shown that by changing the compression type from log compression to 0.07 root compression we are able to outperform MFCCs for all noisy conditions.

CHAPTER I

INTRODUCTION

Nowadays, applications using sound and speech are everywhere. From saying one's account number during a call to the bank's automated system to voice-activated commands on phones and speech-to-text applications. Generally, there are a few important steps that most speech recognition systems undertake before recognizing speech. In these systems, the first step is the feature extraction stage, followed by post-processing and classification stages.

Audition is a highly developed and complicated sense in humans. The human auditory system is capable of localizing, segmenting and recognizing sounds without noticeable degradation in recognition performance in noisy conditions. An average human with normal hearing generally performs well at recognizing minor changes and differences in sounds. Such minor changes most often prove difficult to be quantified by the computer-based pattern recognition systems. One of the major reasons for the performance differences between the human ear and the computerized recognition systems could be due to differences in the type of features extracted by different recognition techniques. Understandably some recognition systems perform better than others. The state-of-the-art recognition features are the mel-frequency cepstral coefficients (MFCCs) that have shown to perform well in audio processing and speech recognition applications. In addition, MFCCs are computationally efficient and easy to implement; however, they do not perform well in noisy signal conditions due to the discarding of temporal information and utilization of triangular filters.

In this work, we study performance of two other types of biologically inspired features, and the effect of changing time constants, window size, compression type, adjacent channel reduction and rectifications types on the overall speech recognition performance using AURORA 2 connected digits data set. The two types of feature sets that we investigate and compare to Mel-frequency cepstral coefficients (MFCCs), are Noise Robust Auditory Features (NRAFs) that are implemented using bilinear transformation, and features similarly extracted using Butterworth filters. We evaluate these features using clean speech and six different SNR levels to evaluate the robustness of features to noise.

CHAPTER II

PREVIOUS WORK

In this work, we investigate robustness of a series of biologically inspired modifications to state-of-the-art features proposed by [11] and [12]. We evaluate our results using a connected digit recognition data set known as Aurora 2.0, and we use the Hidden Markov Model Toolkit (HTK) [20] to determine the percent accuracy of the recognition task.

Aurora 2.0 is a database that contains sets of connected digits consisting of clean speech, as well as noisy speech which have been degraded by different noise conditions at 6 signal-to-noise ratios. The testing data consists of three sets: A, B and C. Set A contains clean speech as well as noisy speech of types subway, babble, car and exhibition hall. Set B contains clean speech in addition to noisy speech of types street, airport, restaurant and train station noise. Set C contains noisy speech of types subway and street noises in addition to clean speech. All noisy speech for all sets are provided at SNRs 20, 15, 10, 5, 0 and -5.

The AURORA database is explained in detail in [10] . This database is designed to help evaluate the performance of speech recognition algorithms under noisy conditions. Aurora has developed standards for Distributed Speech Recognition (DSR) in which the speech analysis is done in the telecommunication terminal and the recognition at a central location in the telecom network. TIDigits database was used as basis to create the Aurora data set. From TIDigits database the data containing isolated digits spoken by US-American male and female adult speakers are used, as well as sequences of up to 7 digits. For the purposes of creating the clean dataset of AURORA, the original data are sampled at 20 kHz

then down-sampled to 8 kHz using an ideal low-pass filter to extract the spectrum between 0 and 4 kHz. Additional filtering is done at this point to take into account the realistic frequency characteristics of the telecommunication terminals. Once additional filtering is applied, some noise is artificially added at different signal-to-noise (SNR) ratios. Noise types are chosen such as to reflect the most likely telecommunication scenarios. The noise types are Suburban train, Babble, Car, Exhibition hall, Restaurant, Street, Airport and Train station. The noises are a combination of stationary, i.e. car and exhibition hall, and non-stationary types, i.e. street and airport. Comparing the noises on a spectral level, the noises appear to be similar, such that the low-frequency region of the long-term spectra seems to have the majority of the signal energy. The noise signals are added to the TIDigits at SNRs 20dB, 15dB, 10dB, 5dB, 0dB and -5dB. Two training modes are defined in [10]. The first is training on clean data only, the second is training on clean and multi-condition data.

For the purposes of this thesis, training is done on clean data only. The advantage of training on clean data only is that speech is modelled without any type of distortion which allows the model to be suited for representing all available speech information. For clean only training set, 8440 utterances from the TIDigits are selected that correspond to 55 adult males and 55 adult females. Signals are then filtered with G.712 characteristics. G.712 refers to one of the standard frequency characteristics defined in ITU [13], in which between 300 and 3400 Hz the curve is flat. For testing, three different sets are defined such that each one has 4004 utterances of 52 males and 52 females that are selected from TIDigits testing set. These 4004 utterances are then split into four subsets, each now including 1001 utterances that have samples from all speakers. Then, one of the noise signal types are added to the utterances in each subset at the SNR levels of 20dB, 15dB, 10dB, 5dB, 0dB and -5dB to create 6 noisy conditions per noise type. The seventh condition is created using clean data only for testing. From the three testing sets, Set A uses four noise types which are suburban train, babble, car and exhibition hall. Set B has restaurant, street, airport and

train station for noise types. Set A and B each have 4 (noise types) x 7 (6 different SNRs + clean) x 1001 utterances = 28028 utterances in total. In addition, in Sets A and B the speech and noise are filtered using G.712 [13] before being added. On the other hand, Set C has only 2 subsets instead of four each with 1001 utterances totalling to 14014 utterances. Also in the case of Set C, the speech and noise are filtered using the MIRS characteristics [13] instead of the G.712. MIRS characteristic refers to one of the standard frequency characteristics defined in ITU [13], in which between 300 and 3400 Hz the curve shows a rising characteristic with an attenuation of lower frequencies [10]. The noise types used for Set C are suburban train and street.

The poor performance of Mel-frequency cepstral coefficients (MFCC) in noisy conditions is due to the reasons that follow. MFCCs are obtained by doing a short-time Fourier Transform to approximate the frequency decomposition along the basilar membrane. The critical bands are modelled using triangular filters placed uniformly along the log frequency axis. Log compression and discrete cosine transform are later used to compress and decorrelate the features [6]. Block processing of the frequency bins results in non-smooth signals in each channel that have lost temporal characteristics. Another contributor to the poor performance of MFCCs under noisy conditions is the use of log compression. This is due to the large negative excursions of the logarithm function for values close to zero, which leads to a spreading of energy in the transform domain after the DCT operation [11]. The use of FFT and triangular filtering in MFCCs lead to discarding non-quantifiable information. In addition, triangular filters are sensitive to small changes in frequency hence they are sensitive to noise.

A better design that is less sensitive to changes in frequency, in which energy estimation is smoother in each channel, is obtained using bandpass filters placed exponentially. In this case, signal strength is estimated using an envelope detector, which is implemented

by a rectifier and a low-pass filter. Since the central auditory neurons do not respond to fast temporal modulations [18], smoothing over 10 ms should not discard relevant information. The use of bandpass filters result in frequency spreading across channels since their design is not very sharp. The spatial derivative, models the lateral inhibitory network in the cochlear nucleus [18]. Applying spatial derivative limits the frequency spreading across channels and results in sharper filter responses. Preliminary study by [11] shows that spatial derivation helps enhance clean and high SNR conditions, but does not help low SNR conditions because it might remove some signal information .

RelAtive SpecTrAl (RASTA) features improve robustness of speech recognition in noisy conditions and its implementation includes critical band analysis, temporal filtering and equal loudness adjustments. The noise robustness comes from filtering out slow changing factors that interfere with speech [5]. The authors in [7] extract an auditory based feature set by a model consisting of cochlear filtering, inner hair cell and lateral inhibition. Furthermore, they only retain the cochlear channel outputs that are more likely to fire the neurons of the central auditory system. The latter features are extracted using principal component analysis (PCA) of non-linearly compressed auditory spectrum. They then evaluate their features using Aurora 2.0 database and show a 40% and 18% improvements in average word error rates compared to the Mel-frequency cepstral coefficients (MFCC) and RelAtive SpecTrAl (RASTA) features, respectively. The authors in [7] have implemented the cochlear filters by using 128 overlapping constant-Q asymmetric bandpass filter, such that their center frequencies are uniformly placed along a logarithmic frequency axis. The inner hair cell (IHC) is implemented by a high-pass filter for fluid-cilia coupling, a sigmoid function for non-linearity of the ionic channel, and a low-pass filter to model the phase-locking decrement in the auditory nerve beyond 2 kHz. The lateral inhibitory network (LIN) of the central auditory system is modelled by a difference operation between adjacent frequency channels followed by a half-wave rectifier to model the non-linearity

of the neurons of the Lateral Inhibitory Network. To model the inability of the neurons in responding to rapid temporal changes, a temporal filtering over a short time window is used. The output of this type of modelling is the auditory spectrum [18]. The authors have replaced the triangular filters of the MFCCs with the auditory spectrum and thereby claim that the noise robustness of their auditory-based features (ABF) is due to the way they have modelled the Inner Hair Cells and the Lateral Inhibitory Network. The phase-locking decrement modelling of the IHC, enhances the signal and the first-order spatial derivative modelling of the LIN reduces the effect of noise [17]. The authors compute the discrete cosine transform (DCT) of the logarithm of the auditory spectrum, keep 13 of the coefficients, and their delta and delta-delta features as well to create a 39-dimensional feature vector. They have used 16 states per digit and 3 Gaussian mixtures per state. For silence they have used three states with 6 Gaussian mixtures per state, and one state short pause model tied to the middle stage of the silence model.

Cui et al. [2] evaluate their noise robust feature extraction algorithms on the Aurora 2 data set. The authors use modified versions of their technique in order to handle SNR mismatch and channel mismatch, and obtain a 47% error rate reduction on Aurora 2 clean training. The methods presented here are for Distributed Speech Recognition (DSR). After the speech versus non-speech detection is done, a Variable Frame Rate (VFR) technique is used in order to choose speech frames. To differentiate between speech and non-speech the input signal goes through a wait-word-pause three state model after the online SNR is determined using log energy. Meanwhile, artificially added white noise is added to the input signal and spectral entropy is calculated afterwards. Then the result is combined with the output from the three-state model in order to determine if the current frame is speech or non-speech. The choice of spectral entropy over energy is that in noisy conditions spectral entropy is more robust hence can act as a complement to energy. For VFR, The authors

use a multiple of 2.5 ms for frame shift and a less than 100 frames per second for an average frame rate. The selection of frame rates are based on the derivatives of weighted log energy Euclidean MFCC distances. In the transition state of speech from consonants to vowels, the Euclidean MFCC distances demonstrate faster changes when compared to the steady-state. The output of the VFR is then run through Fast Fourier Transform, which then the Harmonic Demodulation (HD) is applied to its output in order to reduce the difference between the clean and noisy speech spectrum. The outputs of the HD are then put through a Mel-filter bank to compute their log magnitudes and apply Discrete Cosine Transform (DCT). Peak-isolation and peak-to-valley ratio locking are then applied to the output of the DCT. Finally RASTA filters are applied to the selected frames to produce the noise robust features. Peak isolation is done because in the presence of noise, peaks of the speech spectrum are critical in sound perception. The peak isolation is done by enhancing the peaks by liftering the spectral domain followed by an IDCT to transform back to the spectral domain and then a half-wave rectifier. The peak-to-valley ratio locking, sets the ratio by assigning the highest peak a fixed value of 10 then scaling all other outputs of the Mel-filters.

In [10] the frame rate is set to 100 Hz such that the frame shift is 10 ms. On the recognition side, the HTK software package is used [20]. To model the digits, 16 states per word are used with mixtures of 3 Gaussian mixture models per state. To start the process a vector size of 39 using 12 cepstral coefficients and the logarithmic frame energy plus the corresponding delta and acceleration coefficients. In addition, pause models are defined for silence and pause. The first pause model is called *sil* which models the pauses before and after the utterance and is created using 3 states and 6 Gaussian mixture models per state. The second pause model is called *sp* and is used to model pauses between words and is created using 1 state tied in with the middle state of *sil*.

The authors who created the Aurora front-end in [10] performed recognition using HTK

[20]. The AURORA front-end used for recognition is created by computing 12 Mel frequency cepstral coefficients excluding order 0 and attaching a 13th component, which is the logarithmic frame energy. The 12 coefficients are determined for a speech frame of 25 ms and frame shift of 10 ms. To achieve their cepstral analysis schemes, the authors in [10] also performed offset compensation using a notch filtering operation, as well as a pre-emphasis with a factor of 0.97. In addition, they applied a Hamming window and used a FFT based Mel filter bank with 23 bands ranging from 64 Hz up to half the sampling frequency. In the recognition task, delta and acceleration coefficients are respectively attached to the 13 dimensional Aurora front-end described above, to create a vector of length 39 coefficients. The performance measure for test sets are calculated as the average over all noises and all SNRs between 0 and 20dB (excluding -5dB SNR from the average). The average word accuracy for testing sets A, B and C using clean only training, are 61.34%, 55.74% and 66.14% respectively [10]. The authors note that the recognition accuracy is lower for the noises containing non-stationary segments, which are the noise types babble, restaurant, airport and train, hence the recognition accuracy results of Set B are expected to be lower than that of Set A. Also, Set C which does not include any non-stationary noise, has high accuracy result [10].

In [7] the authors evaluate the robustness of their biologically inspired features for automatic speech recognition derived based on the early processing stages of the human ear. To do so, they replace triangular filters of MFCCs with an auditory model consisting of the cochlear filtering, inner hair cell and lateral inhibition. The authors apply Principal Component Analysis (PCA) to the non-linearly compressed auditory spectrum to extract a feature set that represents the cochlear channel outputs of the neurons within the central auditory system that are most likely to fire. The evaluation is done on the Aurora 2 database and the average error rate improvement is 40% and 18% compared to MFCCs and RASTA features, respectively. The authors focus on the early auditory (EA) processing instead of

a multi-scale cortical representation since the EA is computationally expensive but still robust to noise [7]. In Auditory Based Features (ABF) they replace the triangular filte rbanks of the MFCC with the early auditory processing model proposed [17]. To obtain ABF, the authors compute the discrete cosine transform (DCT) of the log of the auditory spectrum and use the 13 coefficients, and append delta and acceleration coefficients, to create a 39 dimensional feature vector. The authors perform the speech recognition task on the Aurora 2 database using the Hidden Markov Model Toolkit (HTK). They use three types of feature extraction algorithms. First the ABF feature extraction technique is used as described above. Second and third techniques are basically obtained via some type of post-processing of the auditory spectrum. The PCA-ABF applies PCA to the auditory spectrum to get 13 features and append the delta and acceleration coefficients to obtain a vector of size 39. The logPCA-ABF applies the PCA algorithm to the log of the auditory spectrum to find 13 coefficients and later append the delta and delta-delta coefficients to create a 39-dimensional vector. The idea behind using PCA is that the channel outputs of the auditory spectrum that carry that fire neurons carry the most significant information. Using PCA, filter outputs are linearly transformed into a reduced dimension representing the strong components of the spectrum that preserve most of the signal energy. In the case of PCA-ABF, it outperforms MFCCs and ABFs for SNR 10 and below, however for clean speech and SNR above 10 ABF outperforms both [7]. In the case of logPCA-ABF, a non-linearity component, modelling the Outer Hair Cells (OHC) is also added to provide an amplification to signals at low levels. To model the non-linear compression of the OHC, a logarithmic amplitude transformation is used. To apply OHC to the feature extraction process, the logarithm is applied to the auditory spectrum then passed through PCA to obtain logPCA-ABF coefficients. The logPCA-ABF results show improvements for test sets A and B in regards to average word accuracy, as well as the relative word error rate (WER) improvement when compared to MFCCs and RASTA features. The logPCA-ABF shows a 40.2% and 18.71%

relative Word Error Rate improvement over MFCC and RASTA feature performance, respectively. The authors show results for test sets A and B, and exclude results for Set C in which MIRS is used for filtering instead of G.712 as for sets A and B. Hence the result of this feature extraction in response to attenuation of low frequencies as done in set C is unstated and hence pointing to a possible degradation in relative performance.

In [11] the authors propose ways to improve the noise robustness of Mel-frequency cepstral coefficients (MFCCs). The authors use bandpass filters to utilize the temporal information of the signal that is otherwise discarded by triangular filters. In addition, they use a spatial subtraction stage to further improve the performance of the band-pass extracted features. Also, root compression is used instead of log compression in MFCCs, to produce more compact energy values after DCT.

Using the HTK toolkit front-end in [20], the authors use 23 channels to extract MFCCs. Thirteen MFCC coefficients (including the zeroth coefficient) are then mean and variance normalized, delta and acceleration coefficients are appended to create a feature vector of size 39. For the bandpass filters, thirteen features are extracted using the first 13 of the 32 one-sixth octave channels (including the zeroth coefficient) and delta and acceleration coefficients are also appended to create a feature vector of size 39. All coefficients are then mean and variance normalized [1]. The authors show through evaluations that by using a larger root compression (i.e. power of 0.3 instead of 0.07) a better performance is obtained in the case of clean speech; however, more compression can negatively affects the performance in noisy conditions. The authors use a root compression of 0.3 in their analysis regardless of the condition of the signal.

To obtain a smoother energy estimation, the authors in [11] use exponentially spaced band-pass filters and estimate the signal strength using a half-wave rectifier followed by

a low-pass filter. This process is called envelope detection. In order to improve the performance of BPF-MFCC features, the authors add another stage after envelope detection called spatial derivative. To limit the frequency spreading caused by the use of BPFs, spatial derivative is proposed which is implemented as taking the difference between adjacent channels and is motivated by the lateral inhibitory network in the cochlear nucleus. The features produced after this stage are referred to as NRAFs. The spatial derivative also enhances the spectral contrast especially in the case of clean and high SNR conditions. However, in the low SNR cases, the noise variance is equal or greater than that of the signal, hence spatial derivative results in a loss of signal due to subtraction between channels [11] .The authors perform the speech recognition task on the Aurora 2 data set using the Hidden Markov Model Toolkit (HTK) [20]. MFCCs and NRAFs are extracted as explained above and mean and variance normalized (MVN) as detailed in [1]. In the case of 13th coefficient, the zeroth coefficient is used instead of the log energy since it behaves better when subjected to the MVN process. Once features are passed through MVN, the deltas and delta-deltas are appended to form a 39-dimensional vector. To compare the MFCCs and NRAFs the authors use logarithmic compression instead of root compression. The authors show that when 6 Gaussian mixture models are used per word and 12 Gaussian mixture models for silence, the NRAFs outperform MFCCs in all conditions including -5dB SNR, except in clean. As the complexity of the back-end decreases, the MFCCs outperform NRAFs, i.e. 2/4 Gaussian mixture models for words and silence respectively [11]. In addition, in the case of bandpass filters proposed by the authors, features were extracted in a similar manner using 32 one-sixth octave filters. The authors show through preliminary evaluations that by using a larger root compression (i.e. power of 0.3 instead of 0.07) a better performance is obtained in the case of clean speech; however, more compression negatively affects the performance in noisy conditions. That said, the authors use a root compression of 0.3 regardless of the condition of the signal [11].

In [11] a MFCC-type front-end is combined with an auditory based model; however, speech recognition performance does poorly in clean conditions when compared to MFCCs. The authors in [11] later improve their noise robust features in [12] by incorporating varying time constants and gain adaptation. In their previous work, the low-pass filters used in envelope detection models the inability of the neurons in the central auditory system to respond to fast temporal changes; but all the time constants used for the filters are the same. In [12] different time constants are used for different frequency channels to mimic the behaviour shown in neurons such that they have better temporal resolution at higher frequencies and higher frequency resolution at lower frequencies.

A shorter time constant gives better temporal resolution and better performance in higher SNRs. On the other hand, longer time constants produce features that are more noise robust [12]. To evaluate their results, the authors use Aurora 2 data set [10] and the HTK toolkit [20] to do the speech recognition analysis. The authors use 6/12 Gaussian mixture models for each words/silence respectively. The authors show an advantage in using varying time constants to constant time constants, such that NRAFs with varying time constants outperform NRAFs which outperform MFCCs in all conditions except clean [12]. We know that since MFCCs depend on block processing and combination of frequency resolution, the representation has low resolution in time and frequency both. However, in the human auditory system the asymmetrical shape of the cochlear filters result in good time and frequency resolutions, because of its gradual roll-off on the low-frequency side and the sharp cut-off on the high-frequency side [8][12]. In addition, log compression used in MFCCs can negatively affect the performance because it produces large negative values for inputs close to zero causing a spread of energy after the DCT is performed. The use of half-wave rectifier not only has a physiological meaning but it also helps avoid pitch doubling in the lower frequencies from a signal processing point of view [8][12]. In [9] a multi-scale spatio-temporal method is presented that is derived based on the central stages

of the auditory system. Although the approach in [9] presents features that are highly robust to noise in a speech classification task, the method is computationally very expensive.

Since the Hidden Markov Model Toolkit (HTK) is used extensively as the recognition back-end, here we briefly explain its basic principles as described in [20]. HTK is a toolkit that can be used to model any time series by building Hidden Markov Models (HMMs). The main use of HTK is to build HMM-based speech processing tools, especially for speech recognizers. Using training data, HTK training tools estimate the parameters of a set of HMMs then HTK recognition tools are used to transcribe the unknown utterances. In effect, the role of the recognizer is to map continuous speech onto speech vector sequences, then map speech sequences onto the underlying symbol sequences.

In HMM-based speech recognition the assumption is that the sequence of speech vectors corresponding to each word is created by a Markov model. Markov models are finite-state machines that change their states at every time unit. Every time their state changes and moves from state i to a state j a speech vector o_t is created using the probability density $b_j(o_t)$. In addition, the transition from state i to j is governed by a discrete probability a_{ij}. O is known as the observation which is a sequence of speech vectors and a representation of each spoken word. The joint probability that O can be generated by a model M moving through the state sequence X is the product of the transition probabilities (a_{ij}) and output probabilities ($b_j(o_t)$). Since in speech recondition only the observation sequence O is known and the underlying state sequence X is hidden, the model is called Hidden Markov Model.

Since the underlying state sequence X is hidden, the likelihood is calculated by summing over all possible state sequences. This can only be done if the a_{ij} and $b_j(o_t)$ are known. Using an efficient re-estimation procedure in HTK, these parameters can be automatically estimated given a set of training examples. Baum-Welch re-estimation formulae is used to

estimate the parameters. In practice, HTK's HInit tool is used to estimate the initial values of the parameters. First, HInit divides the training vectors equally among the model states and calculates initial values for mean and variance of each state. Then it finds the maximum likelihood state sequence using Viterbi algorithm, then it reassigns the observation vectors to states and does the re-estimation of parameters again. This process is repeated until the estimates do not change. HRest tool works in combination with HInit to construct isolated word HMMs from a set of training examples using Baum-Welch re-estimation. HInit and HRest are used when there is some speech data available such that the location of the sub-word boundaries can be marked and used as bootstrap data. However, when no boot-strap data is available HComV is used instead of HRest and HInit. The no boot-strap data situation is called a flat start in which all phone models are initialized as identical with all state means and variances equal to that of the global speech. After the initial set of models are created, HERest, which is the core of HTK training tool, is used to simultaneously perform a single Baum-Welch re-estimation on the whole set of HMM phone models. This is done such that for every training utterance, the corresponding phone models are concatenated and the forward-backward algorithm is applied to find the means and variances for each HMM in the sequence. Once all the statistics for all training data is processed the re-estimations of the HMM parameters are done using HERest. Since the HTK's philosophy is that HMMs should be refined incrementally, we start with a simple set of single Gaussian context-independent phone models, then iteratively expand them to have context-dependency using multiple mixture component Gaussian distributions. HHed is another tool that is used in combination with HERest. HHEd is used to clone models into context-dependent sets, and to apply parameter tying and to increment the number of Gaussian mixture components. Once training is done, another set of tools are used for testing known as the recognition tools. HVite performs Viterbi-based speech recognition. The inputs to HVite are a list of allowable word sequences, a dictionary and a set of HMMs. HVite converts the word network to a phone network then attaches the appropriate HMM

definition to each phone instance. Recognition is then performed on a list of stored speech files. Once the HMM-based recognizer is built, evaluation of its performance is done using HResults. The evaluation is done by transcribing some pre-recorded test sentences and comparing the output of the recognizer to the correct reference transcriptions. HResults performs the comparison using dynamic programming to align the two transcriptions then counting the number of substitution, deletion and insertion errors to provide the percent accuracy results [20]. s

CHAPTER III

FEATURE EXTRACTION

Some biologically inspired modifications to Mel-frequency cepstral coefficients (MFCC) may improve ASR performance. Ravindran et al. [11] proposed a series of biologically-inspired modifications to MFCCs, and in a preliminary study proved that these modifications improve speech recognition performance. In this work, we perform a more thorough study of the proposed modifications to MFCCs, and investigate the effect of each modification, proposed in [11] on final ASR performance. We hypothesize that some of the biologically inspired modifications to MFCCs may affect the speech recognition results more than others; in addition, we hypothesize that the impact of modifications on final ASR performance may be closely tied to the SNR level of the input signal. The proposed steps for Ravindran's [11] biologically-inspired modifications are shown in Figure 1. We approach our analysis by investigating the role of each modification on final ASR results by experimenting with multiple scenarios for each modification while noting the final speech recognition results for different SNR levels. Our goal is to provide recommendations for SNR-based feature modifications that can improve speech recognition results.

We investigate the effect of modifying filter banks by using band-pass filters instead of triangular filters as used in MFCCs. In two scenarios, we use bilinear transforms and Butterworth filters to construct band-pass filter banks, and to compare the results to MFCC's triangular filters in different SNR levels. We also investigate the effect of varying window sizes, from 8ms to 24 ms, on recognition results to gain insight into how the results vary depending on input SNR levels. In addition, we look into the effect of rectification on final results. In [11] the authors propose half-wave rectification as the first step in envelope

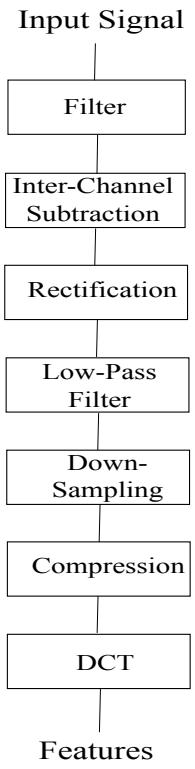

Figure 1: Steps Used to Extract Features

detection. In this work we investigate three rectification scenarios: half-wave rectification, full-wave rectification and squaring. Furthermore, we track the effects of varying time constants used in smoothing of envelope detector on final ASR results. Other modifications we investigate are the effect of spatial derivative. In [11] the authors propose using spatial derivative before envelope detection; however, we investigate the effect of spatial derivative in three scenarios: by placing this step before envelope detection, after envelope detection and removing spatial derivative step all together. The authors in [11] propose a simple downsampling step in which they choose the last member of each frame as a representative of that frame. However, in this work we investigate the possibility of identifying a more

descriptive downsampling method. We compare the result of using three types of features for down-sampling scenarios. The three types of downsampling methods we investigate are for using the average, minimum or maximum of each frame as the representative of that frame. We then look into how the ASR results vary depending on the SNR level and the type of downsampling. Moreover, we look into modifying compression types using log and varying root compressions. In [11] the authors suggest that the results of a 0.07 root compression outperform those of log compression. Here we investigate their claim by using log compression and 0.7, 0.33 and 0.07 root compression to investigate their effect of final ASR levels for all SNR levels. A more detailed description of the analysis follows.

We use two types of filters in our analysis. We then compare the results of the two to the state-of-the-art Mel-frequency cepstral coefficients (MFCCs). We extract the two types of features using bandpass filters that are created by bilinear transforms or Butterworth filters. The sampling frequency in all experiments is set to 8 kHz, since all audio samples belong to the AURORA 2 data set. In the case of bilinear transforms and Butterworth filters, the center frequencies are exponentially placed over 32 channels using $\frac{1}{6}$th octave .

The rational behind using bilinear transforms in implementation of Noise-Robust Auditory Features (NRAFs) [11] is that they can be implemented using available analog hardware. Bilinear transforms, transform continuous-time representations into discrete-time representations. This transform maps every point in the frequency response of a continuous-time filter to a point in the frequency response of a discrete-time filter, such that for every feature in the frequency response of an analog filter there would be a corresponding feature in the frequency response of a digital filter with identical gain and phase shift. However, this mapping is not perfect and the corresponding mapping might end up at a slightly different frequency point in the digital filter. The effect of the mapping mismatch becomes more obvious as frequencies increase and get closer to the Nyquist frequency. To

be more specific, in the continuous-time filter ω_a=0 corresponds to ω =0 in the discrete-time filter; however, $\omega_a = \pm\infty$ in the continuous-time filter corresponds to $\omega = \frac{\pm\pi}{T}$ in discrete-time filter. This non-linear transformation effect is called frequency warping. The bilinear transformation can be constructed such as to reduce the effect of mismatch of frequencies and to compensate for warping. This compensation is called pre-warping. In pre-warping, a matching frequency is specified for which the frequency responses before and after the mapping exactly match. To do pre-warping in MATLAB, the bilinear transformation matches the frequency $2\pi f_{center}$ from the analog s-plane onto the normalized frequency $\frac{2\pi f_{center}}{f_{sampling}}$ in the digital z-plane.

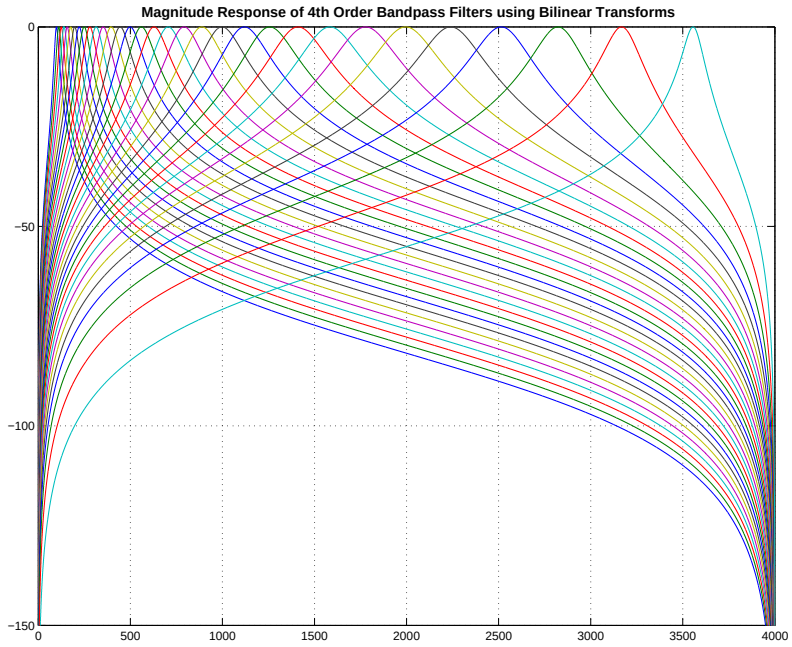

Figure 2: Magnitude Response of 4th Order Bandpass Filters using Bilinear Transforms

The filters used for NRAFs [11] are 32 second-order bandpass $\frac{1}{6}$th octave filters obtained using bilinear transformations, in order to transform from the analog domain onto the digital domain. Pre-warping is done by specifying the center frequencies ranging from 99 hz to 3556 hz. The frequency response of the second-order bandpass filters is set to

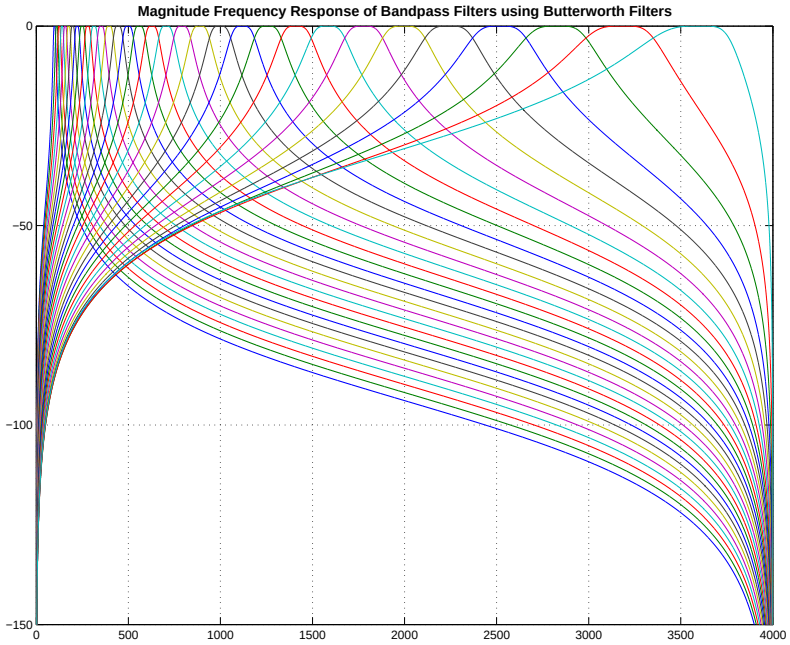

Figure 3: Magnitude Response of 4th Order Bandpass Filters using Butterworth Filters

$\frac{\frac{w}{Q}s}{s^2+\frac{w}{Q}+w^2}$ where Q is the quality factor set to a constant 4.3185 and w is $2\pi f_{center}$. The use of constant Q causes each channel to have the same average power; hence, as the center frequency increase from 99Hz to 3556Hz over the course of 32 channels, the bandwidths also increase accordingly to maintain the constant quality factor. Q $=\frac{f_{center}}{\Delta f}$ where f_{center} is the center frequency of the channel and the Δf (bandwidth) is the width of the range of frequencies for which the total energy is half of the highest energy value. The corresponding coefficients that are generated by the bilinear transformation are then convolved with themselves, which increases the Q factor and makes the filters sharper. Then, using the coefficients obtained from cascaded 4th order pdf the incoming signal is filtered using a one-dimensional filter. Figure 2 shows the corresponding filter banks.

In the case of Butterworth filters, we obtain 4th order bandpass coefficients by passing

in the normalized cut-off frequencies. The lower and higher cut-off frequencies are obtained using $f_{center} \times 2 \times \frac{1}{2^{octave}+1}$ and $f_{center} \times 2 \times \frac{2^{octave}}{2^{octave}+1}$ respectively. A one-dimensional filter is then used on the incoming signal using the coefficients obtained from the Butterworth filter design. The reason for using Butterworth filter over an elliptic or Chebyshev is that Butterworth filters are more monotonic in the passband which for our purposes is more desirable than a steep roll-off. Figure 3 shows the filter banks corresponding to a 4th order bandpass filter created using the Butterworth filters, as described above.

Figure 1 shows the steps required for extracting the biologically inspired features proposed in [11]. The block "Filter" in Figure 1 refers to band-pass filters created by either bilinear transforms or Butterworth filters. Thirty two one-sixth octave channels are used while filtering. In each channel the output of the filter is subtracted from the output of the channel immediately proceeding. This step is called the inter-channel subtraction or spatial derivative. This is followed by rectification and a low-pass filter to detect the envelope of the signal. The result of the envelope detector is then down-sampled for each channel. The resulting matrix including the features for all 32 channels is compressed and a discrete cosine transform is applied to decorrelate the coefficients. From the resulting matrix the features corresponding to the first 13 channels are chosen as the 13 coefficients. All coefficients are then mean and variance normalized (MVN) using the method in [1]. In the case of 13th coefficient, the zeroth coefficient is used instead of the log energy since it behaves better when subjected to the MVN process [11]. The delta and acceleration coefficients are then added, using window sizes of 3 and 2 respectively, to create 39-dimensional feature sets. The method to extract MFCCs are that of in [11], in which 23 channels are used to extract MFCCs.Thirteen MFCC coefficients (including the zeroth coefficient) are then mean and variance normalized, delta and acceleration coefficients are appended to create a feature vector of size 39. It is important to note that rectification and low-pass filter steps together are called envelope detection.

In our analysis, first we create bandpass filters using bilinear transforms with varying window sizes from 8ms to 24ms to study the effect of window size on the overall results. This was done for octave $\frac{1}{6}$th using the three types of rectification: half-wave, full-wave or squared, and two Alphas for smoothing in the low-pass filtering step of the envelope detector. The parameter used in the low-pass filter, Alpha, is set such that the effect of more smoothing (Alpha 1) versus less smoothing (Alpha 2) is investigated based on varying time-constants, which are tied to the center frequency of the channels. The values for Alpha 1 and Alpha 2 are listed in Table 1. For down-sampling, we investigate the effect of using minimum, maximum and mean as representative of each frame.

Following that, we change log compression to varying root compression starting at 0.01 with incremental increases of 0.01 up to 0.09. We used this compression on bandpass filters created using bilinear transforms, 10 ms window size, full-wave rectification and high smoothing (Alpha 1). Root compressions of 0.7 and 0.33 were also evaluated but the results were not as good as the results of root compression for 0.01 through 0.09.

On the recognition back-end, we use 20 states per digit and 3/6 Gaussian mixtures for each word/silence. For silence we use three states with 6 Gaussian mixtures per state, and one state short pause model which is tied to the middle stage of the silence model. Only clean files are used for training, however the testing includes clean and noisy data as per Aurora 2 database explained in detail in chapter 2. The performance measurements are reported as the "percent accurate" for the test sets and are calculated as the average over all conditions, including all clean and noisy data. On the recognition back-end we have utilized the Hidden Markov Model Toolkit (HTK) [20].

Since no boot-strap data is available in our training task, HCompV is used to create

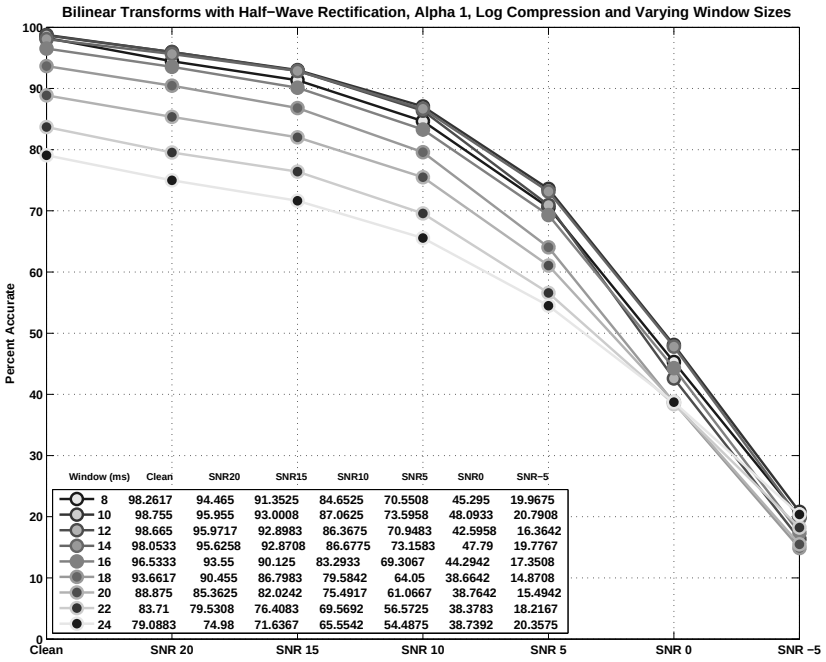

Figure 4: Bilinear Transforms with Half-Wave Rectification, Alpha 1, Log Compression and Varying Window Sizes

an initial set of models. Following that, HERest is used for embedded training using the entire training dataset. HHEd is used in combination with HERest to make the models context-dependent, increase the number of Gaussian mixtures from 1 to 3 and to tie in the parameters. In other words, the HMMs are trained in stages first using HHEd to modify the HMMs then using HERest to re-estimate the parameters of the modified set, going back and forth until all 20 states are completed. Once training is completed, testing is done on sets A, B and C. For each set, first HVite builds the HMM recognizer for each of the 4 file sets within testing set A and B, as well as each of the two file sets within testing set C. Once the HMM-based recognizer is built, evaluation of its performance is done using HResults. The percent accuracy results given by HResults for each test set are used to calculate the average accuracies used in our result evaluations.

In our analysis provided below, it is important to note that our default settings are as

follow: octave is $\frac{1}{6}$th. Bandpass filter order using bilinear transform is 4. Butterworth filter order is 4. Window size is 10ms. The spatial derivative is performed before the rectification step. Rectification is full-wave. The alpha used for low-pass filtering is Alpha 1 which implies more smoothing (values are listed in Table1). Down-sampling is done by choosing the last element of each frame as the representative of that frame. Compression type is log unless root compression is noted. All these settings apply unless otherwise explicitly noted.

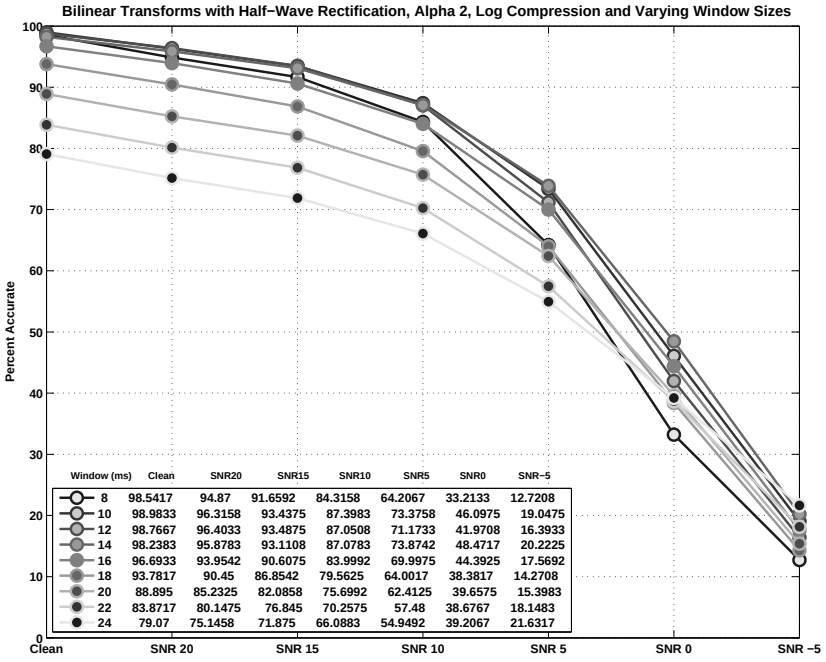

Figure 5: Bilinear Transforms with Half-Wave Rectification, Alpha 2, Log Compression and Varying Window Sizes

3.1 Window Sizes

One of the modifications to MFCCs that we investigate is the effect of varying frame size on final ASR results. The window sizes are increased in steps of 2 ms starting from 8 ms up to 24 ms. This is done for bilinear transforms in the case of log compression with octave

$\frac{1}{6}$th, for rectification types of half-wave and full-wave for Alpha 1 (high smoothing) and Alpha 2 (low smoothing) and log compression. It was determined that window size of 10 ms generally provides the best overall results. A biological explanation could be that according to [18] the central auditory neurons cannot respond to fast temporal modulations, hence smoothing over 10 ms seems to be a biologically sound window size. However, another possible reason that 10 ms window size produces better results can be due to the speech recognizer set-up. According to [20], in order to form an exact waveform representation using discrete sequence of parameter vectors, which have been transformed from continuous speech, a 10 ms window can make the waveform be regarded as stationary; hence, a more exact sequence is mapped from continuous speech using the HTK recognizer. For the remainder of our analysis, we use 10 ms as our window size of choice, since it shows the best overall results when compared to other window sizes.

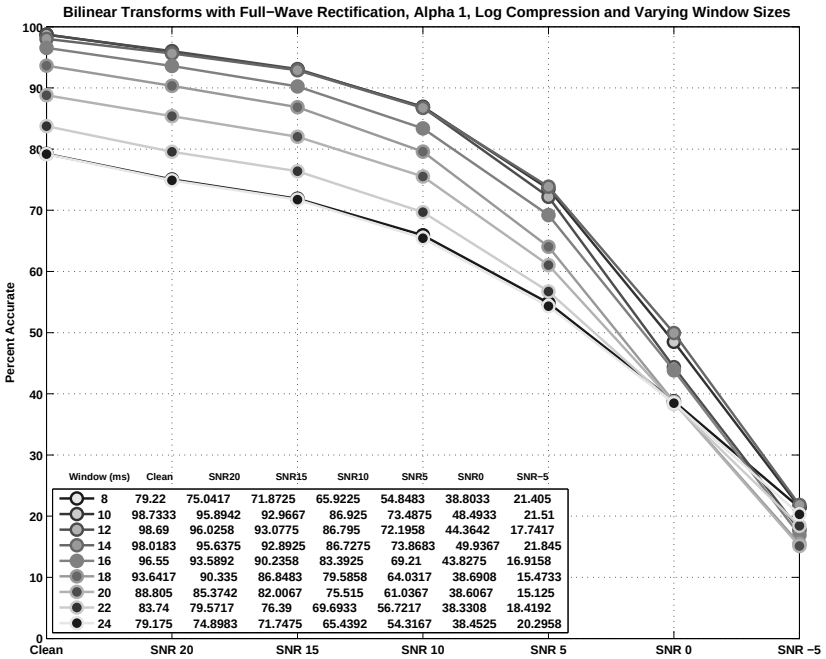

Figure 6: Bilinear Transforms with Full-Wave Rectification, Alpha 1, Log Compression and Varying Window Sizes

It is observed that as the window size increases, the performance in low SNRs improves yet performance in clean speech degrades. This can be explained, because as the window size increases, temporal information decreases. Temporal information are important especially in noisy conditions since there will be less information to draw on. Figure 4 and Figure 5 show how the speech recognition results change as a result of changes in window sizes for half-wave rectifiers with Alpha 1 and Alpha 2. It is important to note that Alpha 1 provides more smoothing compared to Alpha 2, hence the results in noisy conditions are expected to be better. Figure 6 and Figure 7 show the results for Alpha 1 and Alpha 2, respectively, when full-wave rectification is used. It can be seen from these figures that regardless of the rectification or smoothing, as the window size increases the result for clean degrades, except in the case of 8 ms window; in addition, as the window size increases the results for the noisy conditions fluctuate and hence are inconclusive.

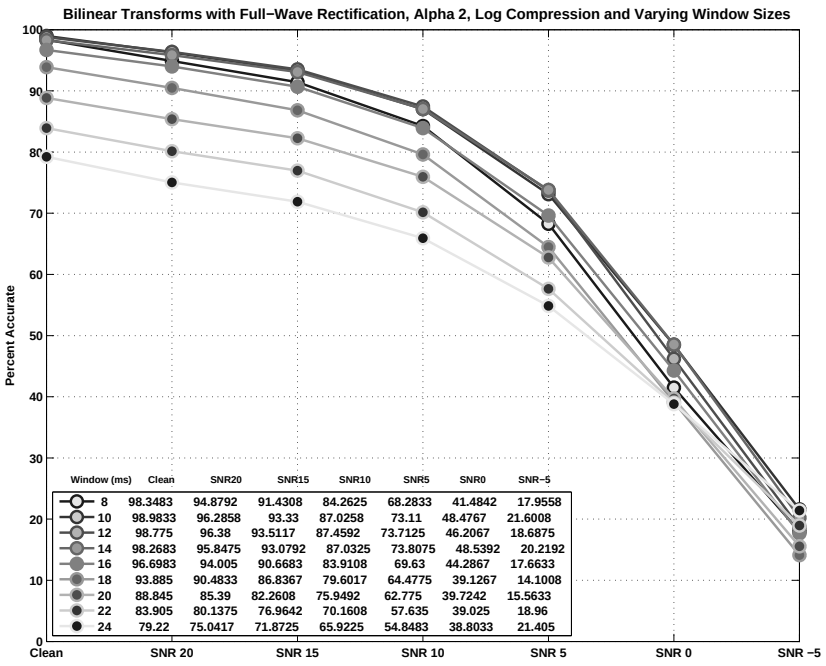

Figure 7: Bilinear Transforms with Full-Wave Rectification, Alpha 2, Log Compression and Varying Window Sizes

3.2 Spatial Derivative

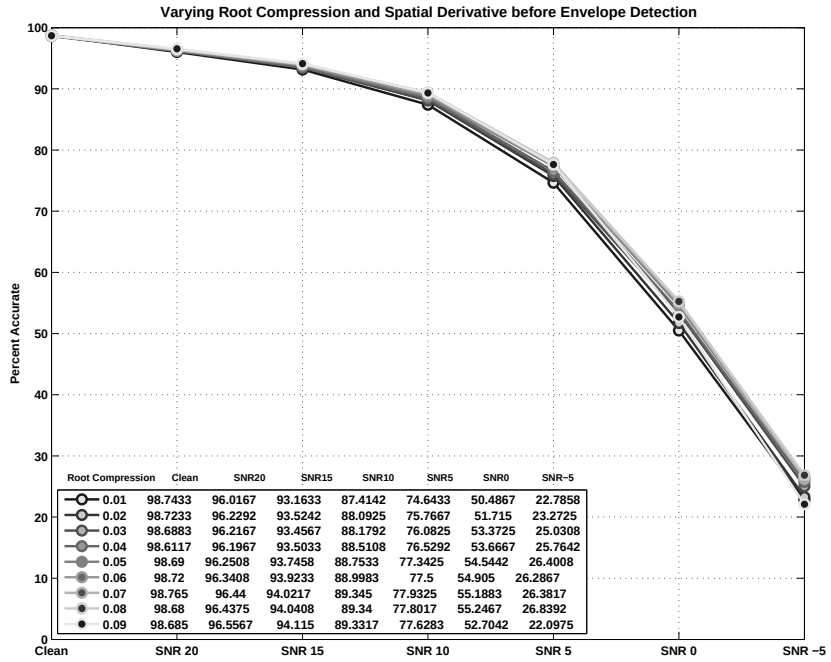

Figure 8: Bilinear Transforms with Window Size 10ms Full-Wave Rectification, Alpha 1, Varying Root Compression and Spatial Derivative Before Envelope Detection

One of the biologically inspired modifications proposed in [11] is to place a spatial derivative or inter-channel subtraction step before envelope detection. According to [11] the spatial derivative is done to improve the performance in higher SNRs by removing the wide-band noise and reducing inter-speaker variability. However, in low SNR cases removing the spatial derivative produces better results since the subtraction might be eliminating some useful data. The use of spatial derivative is to limit the spreading of frequency between channels resulting in sharper filter responses, as well as to enhance the spectral contrast. Here we investigate the effect of spatial derivative in three ways. First, we apply spatial derivative to all channels *before* the envelope detection steps. Second, we apply it to all channels *after* the envelope detection steps and before down-sampling. Third, we remove spatial derivative all together. These scenarios are studied for bilinear transforms with varying root compressions, $\frac{1}{6}$th octave, full-wave rectification and Alpha 1 in the low-pass

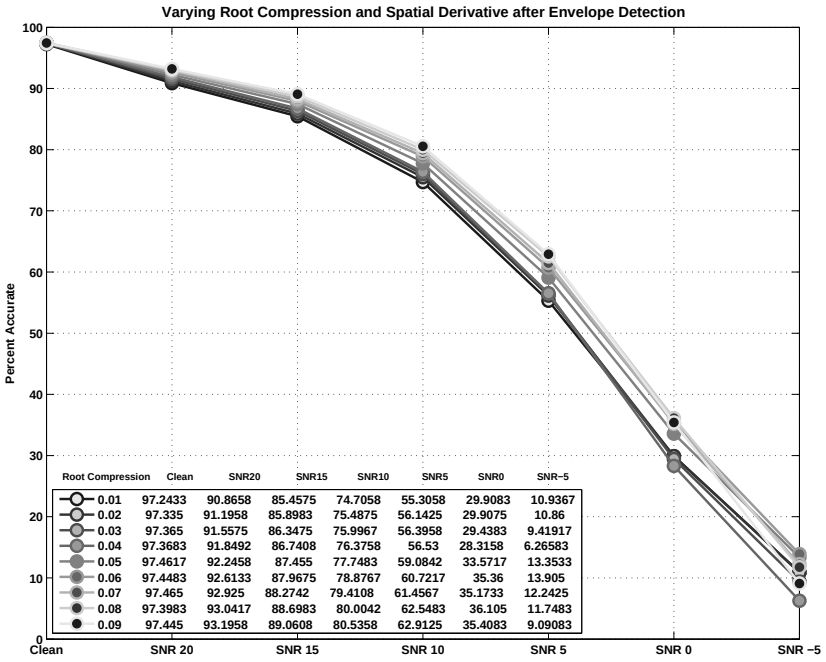

Figure 9: Bilinear Transforms with Window Size 10ms Full-Wave Rectification, Alpha 1, Varying Root Compression and Spatial Derivative After Envelope Detection

filter. Figure 8, Figure 9 and Figure 10 show the three cases respectively. Figure 11 shows the effect of applying spatial derivative to bandpass filters created using bilinear transforms using 10 ms frame size, 0.07 root compression, full-wave rectification and Alpha 1. Root compression 0.07 is chosen because it yields the best overall results for all conditions.

From Figure 11 it can be seen that when spatial derivative is applied *after* the envelope detection step the results are poor. On the other hand, in cases of clean speech, SNR20, SNR15 and SNR0 the best results are achieved when no inter-channel subtraction is applied. The case in which spatial derivative is not applied follows very closely to the scenario in which spatial derivative is applied prior to the envelope detection. However for SNR10, SNR5 and SNR-5 the best results are found when inter-channel subtraction are applied before the envelope detection. This result is unexpected, since the spatial derivative has been claimed to have been useful especially in the case of clean speech according to

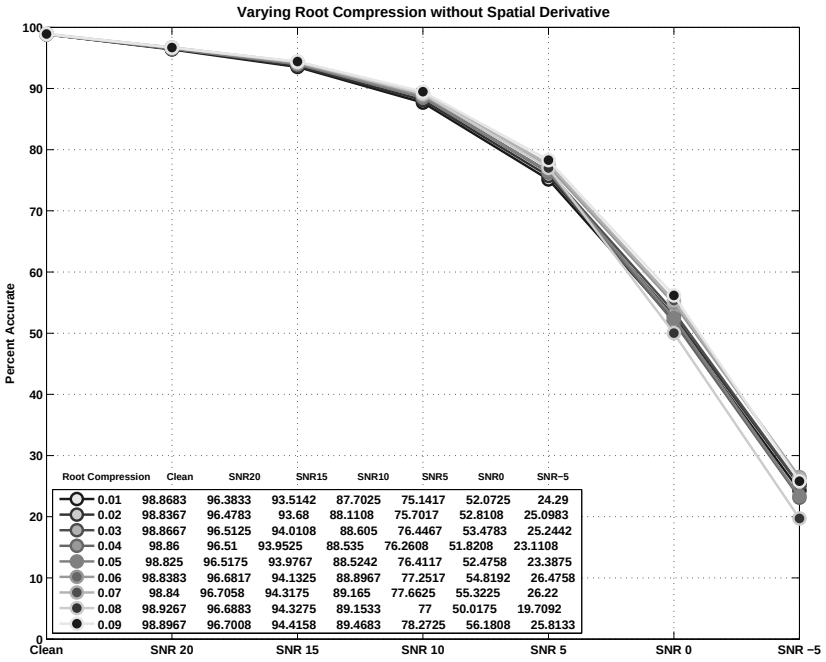

Figure 10: Bilinear Transforms with Window Size 10ms Full-Wave Rectification, Alpha 1, Varying Root Compression without Spatial Derivative

[12]. However, it must be noted that any changes in other parameters and in HTK training could very much affect the final results. For instance in [12] the number of Gaussian mixtures used is 6/12 per word/silence while training 38 HMMs. Whereas in our model we use 3/6 Gaussian mixtures per word/silence while training 20 HMMs. In other words, using a more complex back-end can impact the final results more than using the spatial derivative or not using it, hence spatial derivative does not appear to be a strong influencer of final ASR results.

3.3 Types of Rectification in the Envelope Detector

Rectification is the first step in detecting a signal's envelope. One of the modifications proposed in [11] is to apply half-wave rectification before passing the signal through a low-pass filter for smoothing. To investigate the role of this modification on final ASR results, we use three types of rectification: half-wave, full-wave and squaring of the signal.

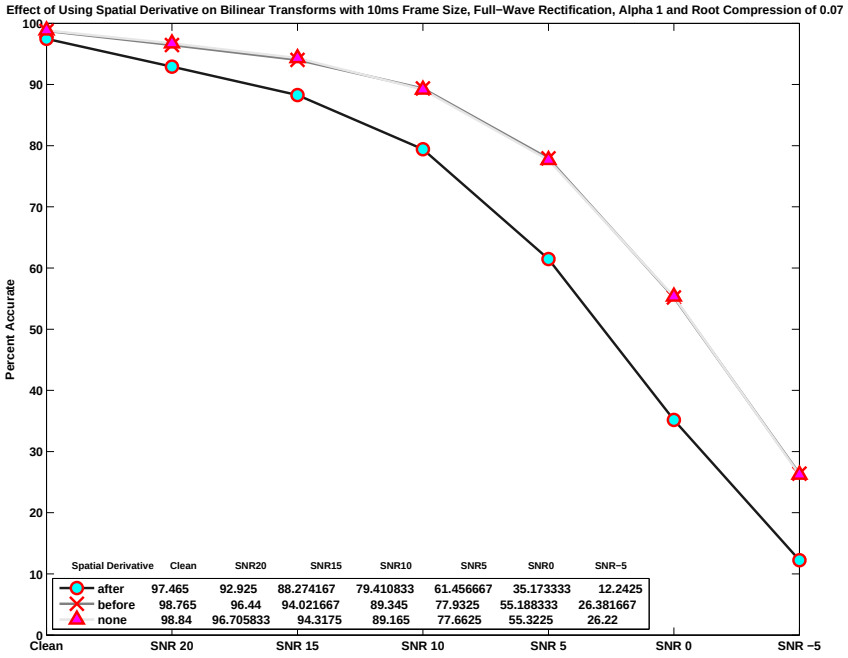

Figure 11: Effect of Applying Spatial-Derivative

Full-wave rectification results in the best outcome in default settings for very low SNRs; however, half-wave rectification shows the best results for all other cases. Squaring has the least desirable outcome in this case. This can be justified by saying that squaring of a signal results in product terms which might be very low in frequency, yet close to that of the envelope. Since these product terms are large relative to the envelope, they might be distorting and overpowering the envelope, resulting in a relatively lower performance. Also the product terms might cause strange harmonic product terms compared to the absolute value for the full-wave and half-wave rectifiers but its likely when you do the squaring you get the harmonics of the product terms resulting in the energy being spread over a larger frequency range, most of which is filtered out. Therefore, the full-wave or half-wave rectification is likely to have less distortion when dealing with a waveform with multiple harmonics. Full-wave rectifier provides better results in very noisy conditions since it can follow the signal better as compared to half-wave and squaring.

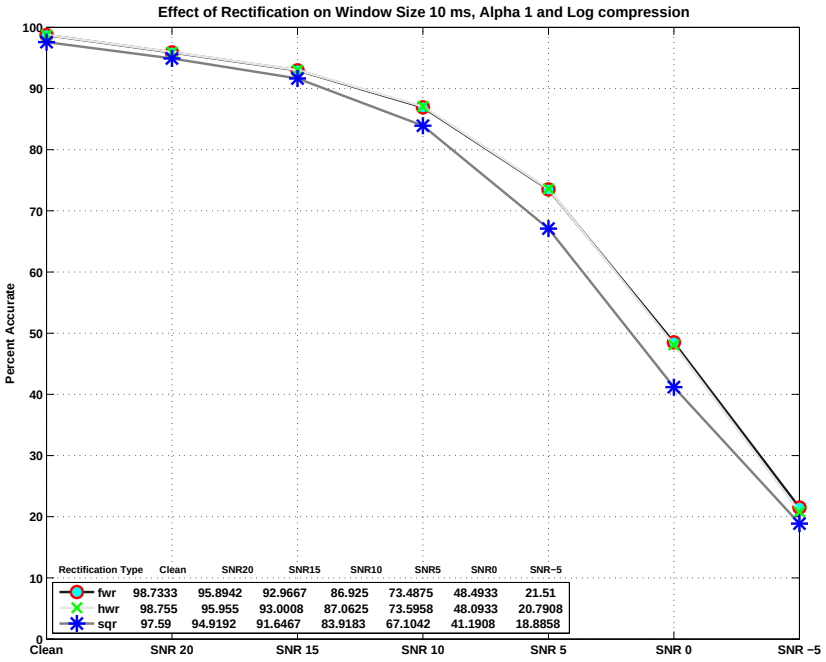

Figure 12: Effect of Applying Three Types of Rectification in the Envelope Detector

Figure 12 shows the effect of using the three types of rectification described above in the envelope detection stage of the process. The results shown in the Figure 12 are for the cases of full-wave, half-wave and square with window frame size of 10 ms, Alpha 1 and 0.07 root compression.

3.4 Low-Pass Filter Parameter in the Envelope Detector

Smoothing is the second step in the envelope detection. In [11] the authors propose Alpha 1 as one of the modifications that improve the overall ASR results. Here we investigate the effect of smoothing on speech recognition results in different SNR using two sets of Alpha values. Alpha represents the coefficient used in the low-pass filter. Alpha 1 and Alpha 2 are calculated by $\exp^{(}\frac{-1}{8\times timeconstant_{(channel)}}$) and $\exp^{(}\frac{-1}{2\pi\ \times timeconstant_{(channel)}}$) respectively. As a result, Alpha 1 provides more smoothing compared to Alpha 2. More smoothing creates superior results in noisy conditions over clean. Hence Alpha 1 results are superior to those

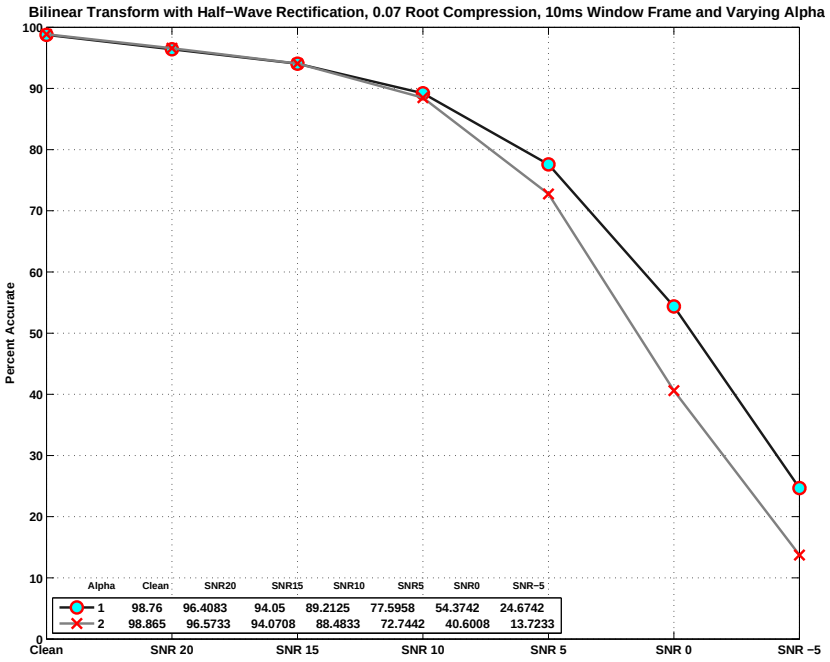

Figure 13: Bilinear Transform with Half-Wave Rectification, 0.07 Root Compression, 10ms Window Frame and Varying Alpha

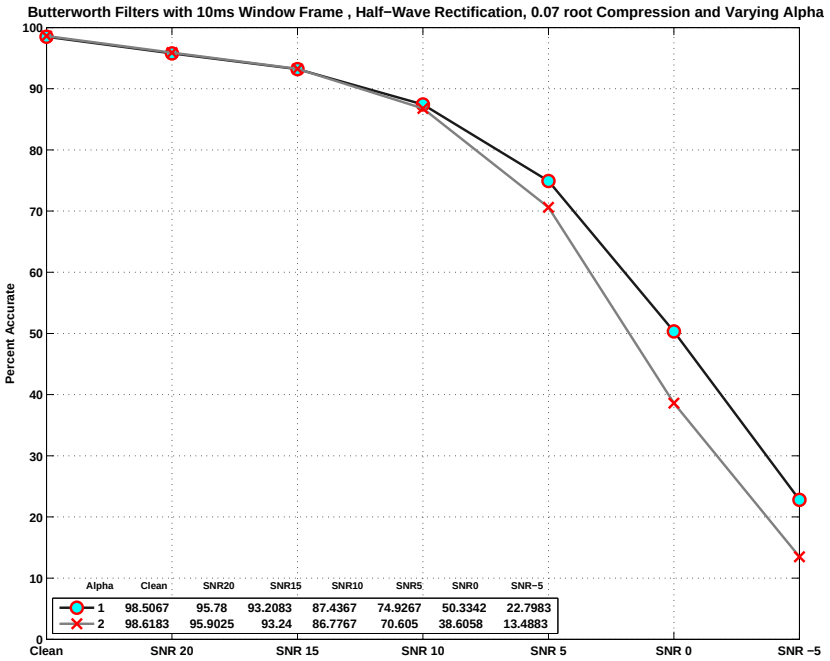

Figure 14: Butterworth Filters with Half-Wave Rectification, 0.07 Root Compression, 10ms Window Frame and Varying Alpha

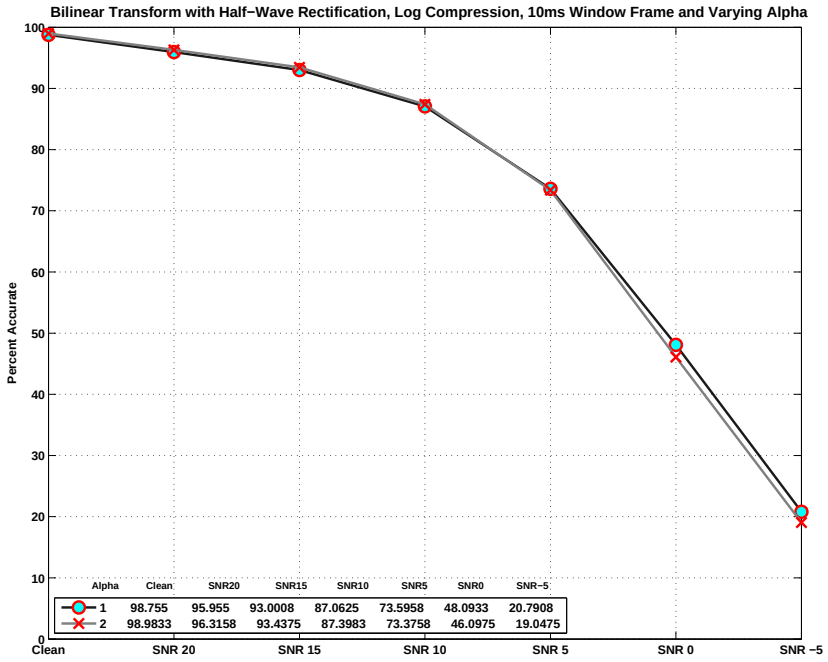

Figure 15: Bilinear Transform with Half-Wave Rectification, Log Compression, 10ms Window Frame and Varying Alpha

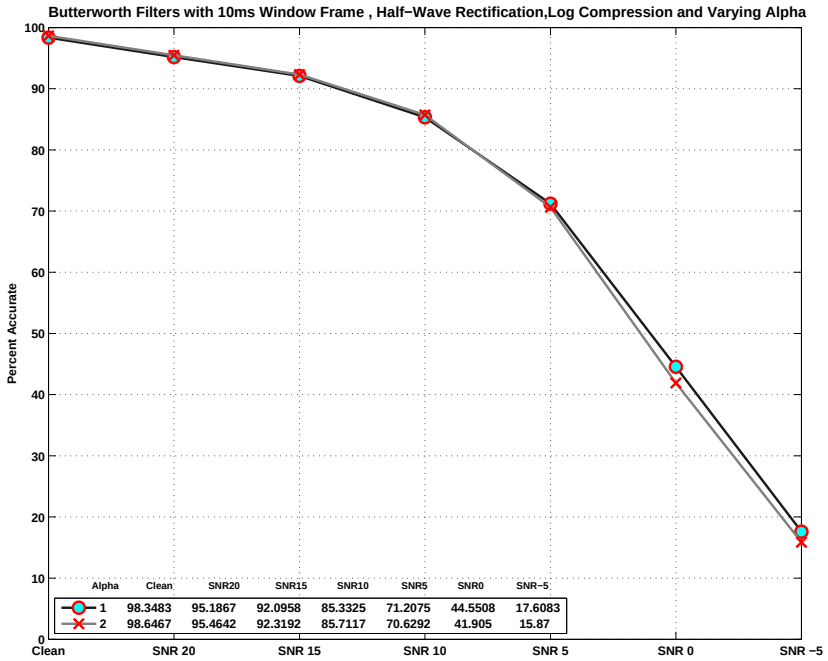

Figure 16: Butterworth Filters with Half-Wave Rectification, Log Compression, 10ms Window Frame and Varying Alpha

of Alpha 2's in noisy conditions, and Alpha 2 results are superior to those of Alpha 1's in clean speech and less noisy conditions. Table 1 lists all values for time constants and Alphas in all 32 channels.

Figure 13 and Figure 14 show the results of varying Alpha for bandpass filters using bilinear transforms and Butterworth filters, respectively, with 10 ms window size, half-wave rectification and 0.07 root compression. Similarly, Figure 15 and Figure 16 show the same type of results but with log compression instead of root compression.

According to [12] shorter time constants provide better temporal resolution and enhanced performance in clean speech and high SNR conditions. On the other hand, longer time constants provide more smoothing hence features that are more noise robust as compared to the shorter time constants. Equation 1 shows the method used for calculating the time constants [12] . In this equation, tc, fs and tc represent Time Constant, Sampling Frequency and Center Frequency, respectively . The time constant and Alpha values change depending on the center frequency of the channel. It can be seen from Table 1 that as the channel numbers increase, the time constants decrease and so do the Alpha values. In other words, as the channel number increases less smoothing is done.

$$tc = (\frac{18.4}{fs}) \times (\frac{fs}{2} - fc) + 31 \tag{1}$$

From these four figures, it can be seen that ASR results corresponding to Alpha 1 are generally superior for SNR5, SNR0 and SNR-5, and the results generated by Alpha 2 are superior for all other cases.

Table 1: Time Constants and Alpha Values

Channel Number	Center Frequency	Time Constant	Alpha 1	Alpha 2
1	99	39.9723	0.9969	0.996
2	111.1	39.9444	0.9969	0.996
3	124.7	39.9131	0.9969	0.996
4	140	39.878	0.9969	0.996
5	157.2	39.8385	0.9969	0.996
6	176.4	39.7943	0.9969	0.996
7	198	39.7446	0.9969	0.996
8	222.2	39.6888	0.9969	0.996
9	249.5	39.6262	0.9969	0.996
10	280	39.556	0.9968	0.996
11	314.3	39.4771	0.9968	0.996
12	352.8	39.3886	0.9968	0.996
13	396	39.2892	0.9968	0.996
14	444.5	39.1777	0.9968	0.9959
15	498.9	39.0525	0.9968	0.9959
16	560	38.9119	0.9968	0.9959
17	628.6	38.7542	0.9968	0.9959
18	705.6	38.5771	0.9968	0.9959
19	792	38.3784	0.9967	0.9959
20	889	38.1553	0.9967	0.9958
21	997.9	37.9049	0.9967	0.9958
22	1120.1	37.6239	0.9967	0.9958
23	1257.2	37.3084	0.9967	0.9957
24	1411.2	36.9543	0.9966	0.9957
25	1584	36.5568	0.9966	0.9957
26	1778	36.1106	0.9965	0.9956
27	1995.7	35.6099	0.9965	0.9955
28	2240.1	35.0477	0.9964	0.9955
29	2514.4	34.4168	0.9964	0.9954
30	2822.4	33.7086	0.9963	0.9953
31	3168	32.9136	0.9962	0.9952
32	3556	32.0213	0.9961	0.995

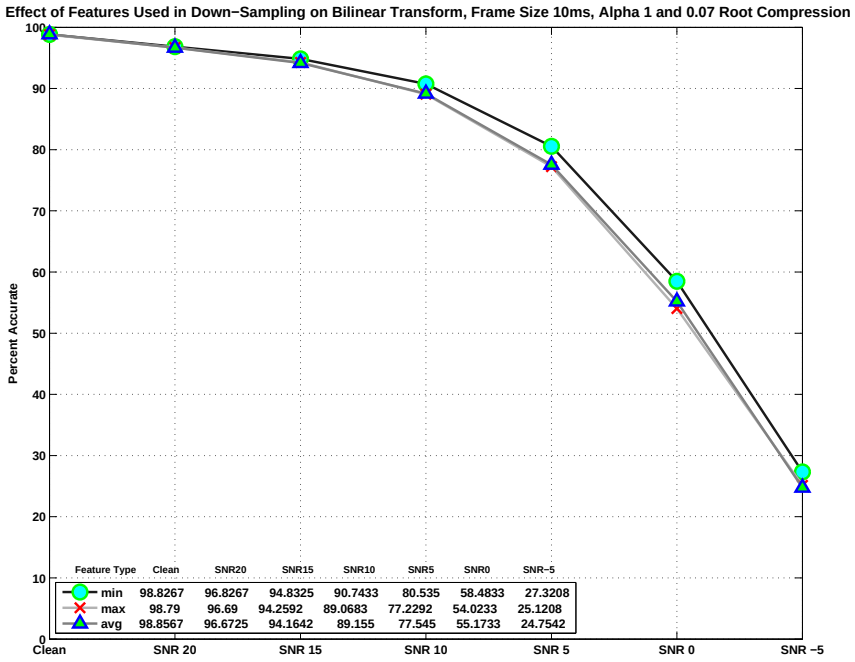

Figure 17: Effect of Down-Sampling

3.5 Down-Sampling

In [11] down-sampling of each frame is done by simply choosing the last element of each frame as a rough representation of that frame. The down-sampling step is done after envelope detection to reduce the size of the feature vector. To do a more intelligent down-sampling and investigate the effect of this modification on final ASR results, we perform down-sampling using three methods: by using the maximum , minimum or mean of each frame to represent that frame. We perform our analysis using bilinear transformation for 0.07 root compression with full-wave rectification, Alpha 1 and a 10 ms window size. Figure 17 shows that using the minimum value results in the highest recognition accuracy for all conditions except for clean speech. For clean speech the mean value produces the best results.

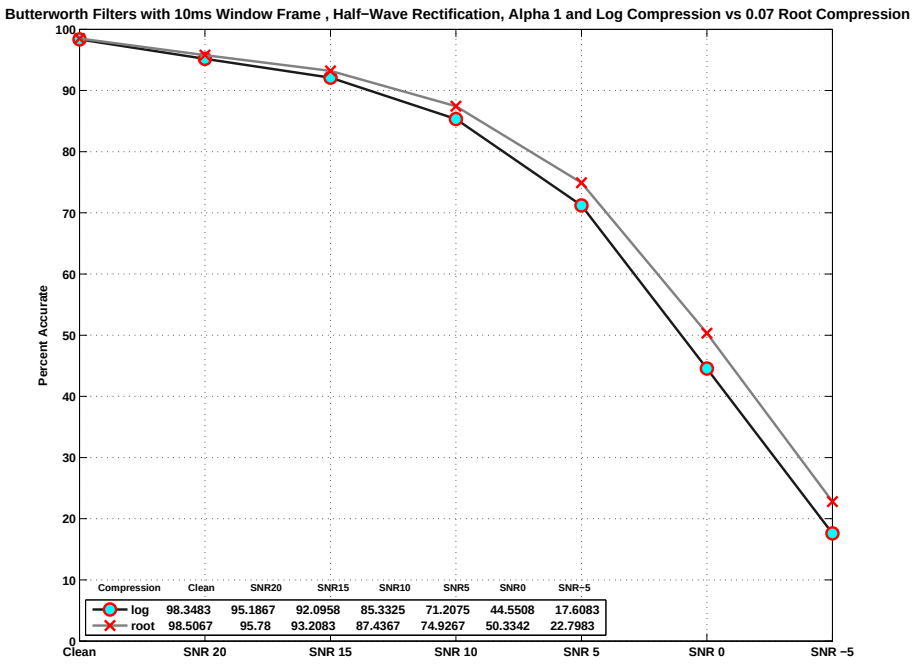

Figure 18: Effect of Root vs Log Compression for Butterworth filter with Window 10ms, Half-Wave Rectification and Alpha 1

3.6 Compression

One of the biologically inspired modifications made to MFCCs, proposed in [11] and [12], is the use of root compression instead of log compression. Here we study the effect of this modification on speech recognition results with varying input SNR. The effect of root compression versus log compression is studied for the case of 4th order Butterworth filters with half-wave rectification, Alpha 1 and Alpha 2. These results are shown in Figure18 and Figure19 for the cases of Alpha 1 and Alpha 2, respectively.

In the case of Alpha 1, where more smoothing is done, Figure18 shows that the root compression outperforms the log compression in all SNR conditions. However, when less smoothing is done with Alpha 2, Figure19 shows that 0.07 root compression outperforms log compression only for clean speech and SNR5 and below. This shows that to achieve the best results, an interplay of a combination of different parameters is required. For instance the amount of smoothing used could be more significant in the overall results than the type

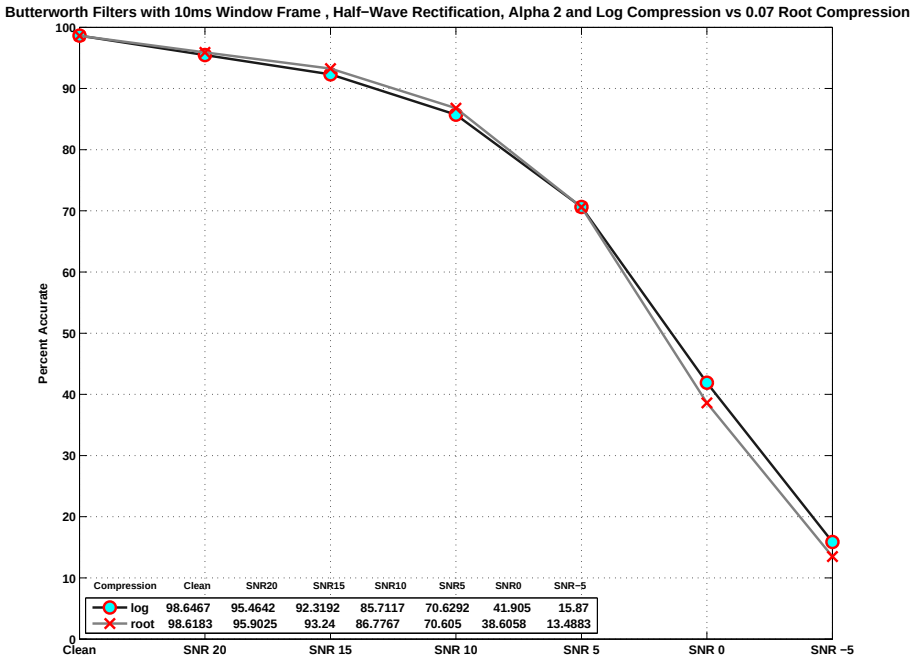

Figure 19: Effect of Root vs Log Compression for Butterworth filter with Window Size 10ms, Half-Wave Rectification and Alpha 2

of rectification used. It is observed that when rectification is changed from half-wave to full-wave in the case of Alpha 1 for Butterworth filters, half-wave rectification still outperforms full-wave rectification in all cases except for clean speech. Also, it is important to note that when more smoothing is done, root compression outperforms log compression, but when less smoothing is done the type of compression used can vastly affect the final results.

To determine an appropriate root compression factor, we examine 0.7, 0.3, 0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02 and 0.01 as root compression factors. We determine that root compressions higher than 0.1 produce very poor results, hence the root compression factors 0.01 through 0.09 are further investigated. More analysis shows that root compression factor 0.07 produces the best overall results. Figure 20 shows the recognition results

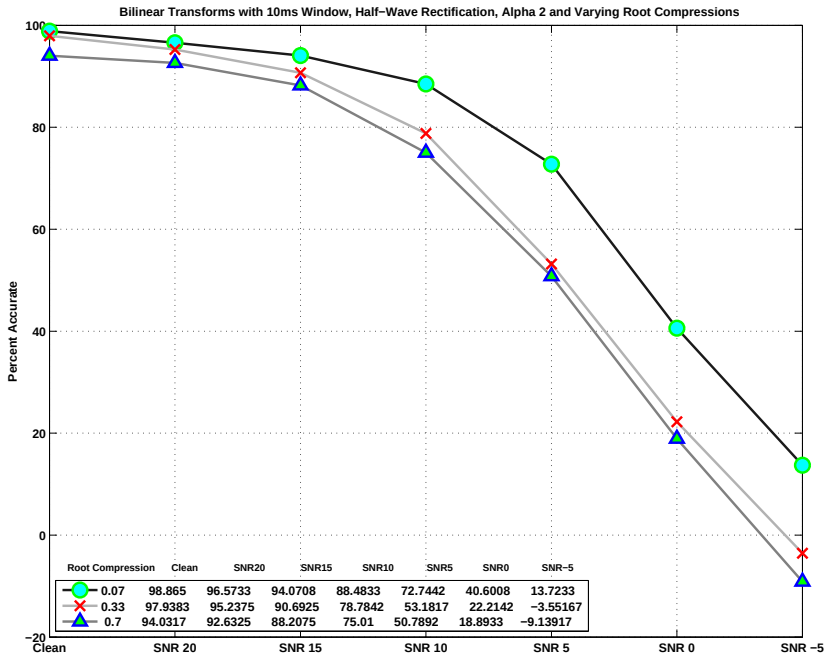

Figure 20: Effect of Varying Root Compression

for bandpass filter banks created using bilinear transforms using window size 10 ms, half-wave rectification, Alpha 2 and root compressions 0.7, 0.33 and 0.07. It can be seen from Figure 20 that by reducing root compression from 0.7 to 0.33 then to 0.07 the recognition results show improvements for clean speech as well as for all SNR cases.

Comparing the performance of bandpass filters extracted using bilinear transforms with full-wave rectification and Alpha1, when the compression type changes from root as in Figure 23 to log as in Figure 22, it can be seen that these filters perform superior when root compression is used rather than log (case of bi-log versus bi-root in the two figures) in all cases. However, as shown in Figure 21 log compression performs better for clean speech and for all other cases root compression results are superior. In addition comparing Figure18 to Figure19 it can be seen that the best results for clean speech is for log compression in Alpha 2, but in all other cases root compression outperforms log results. Hence,

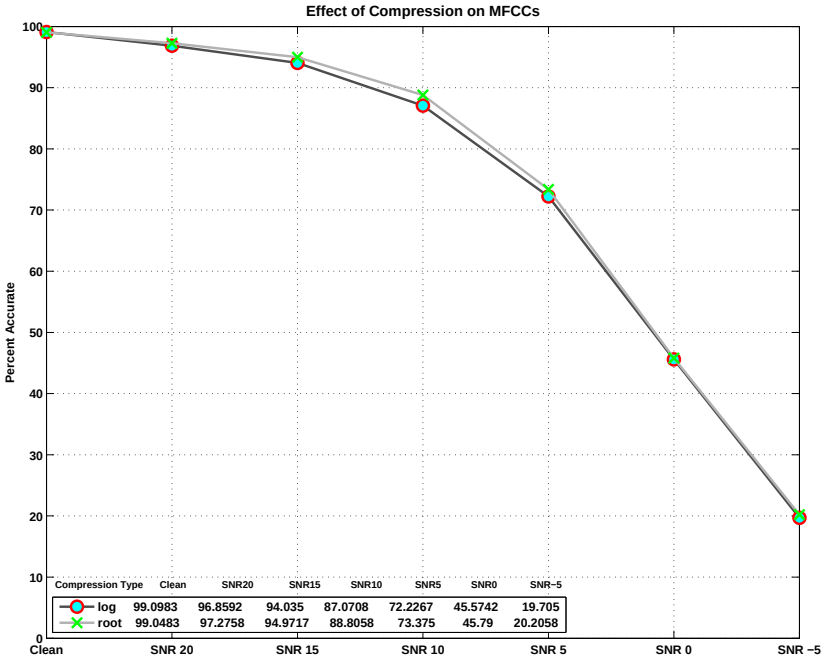

Figure 21: MFCCs

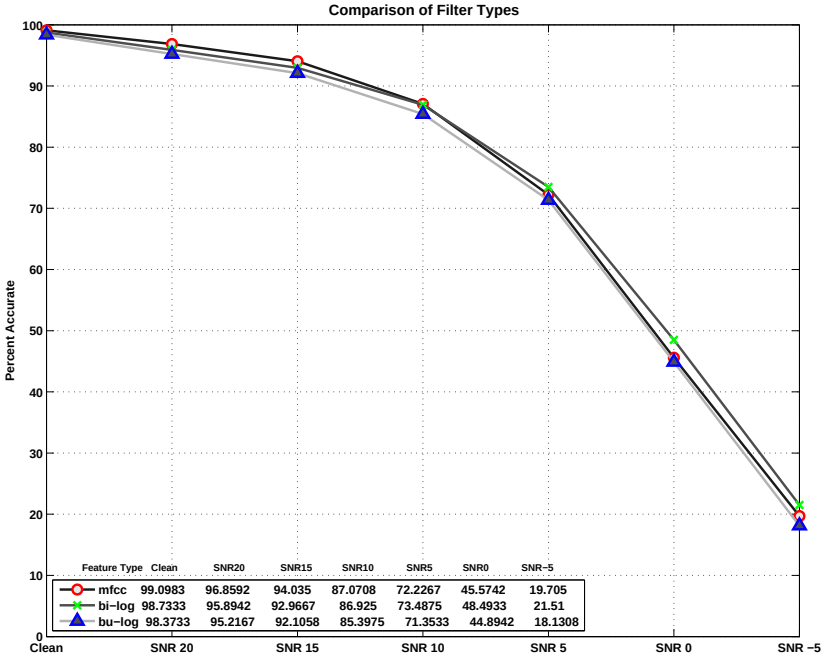

Figure 22: MFCC vs Bilinear Transforms and Butterworth Filters with Log Compression

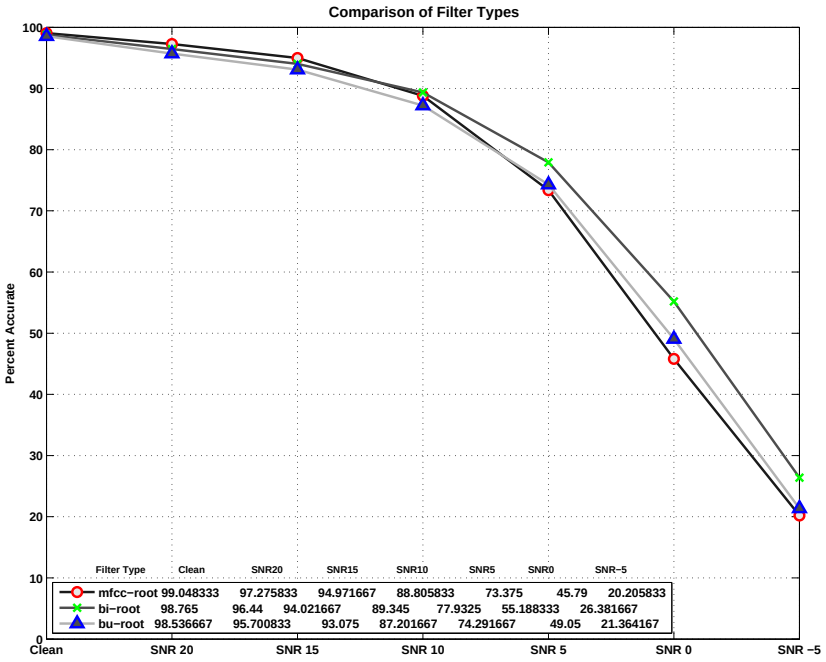

Figure 23: MFCC vs Bilinear Transforms and Butterworth Filters with Root Compression

it is concluded that in general log compression produces superior results for clean speech and root compression does the same for all other conditions. This could be explained by looking at the values before compression in noisy speech. In noisy speech there are more values that are close to zero. We know that performing log operation on a number close to zero results in a large negative excursion that lead to a spreading of energy in the transform domain after the DCT operation [11]. Whereas root compression does not cause in negative excursions, hence it outperforms log compression in noisy conditions. Since clean speech generally does not result in too many close to zero values before compression, log compression performs well.

Figure 22 and Figure 23 show the comparison of results for log versus 0.07 root compression, respectively for features extracted using MFCCs, bandpass filters using bilinear transforms and Butterworth filters with full-wave rectification and Alpha 1. It can be seen

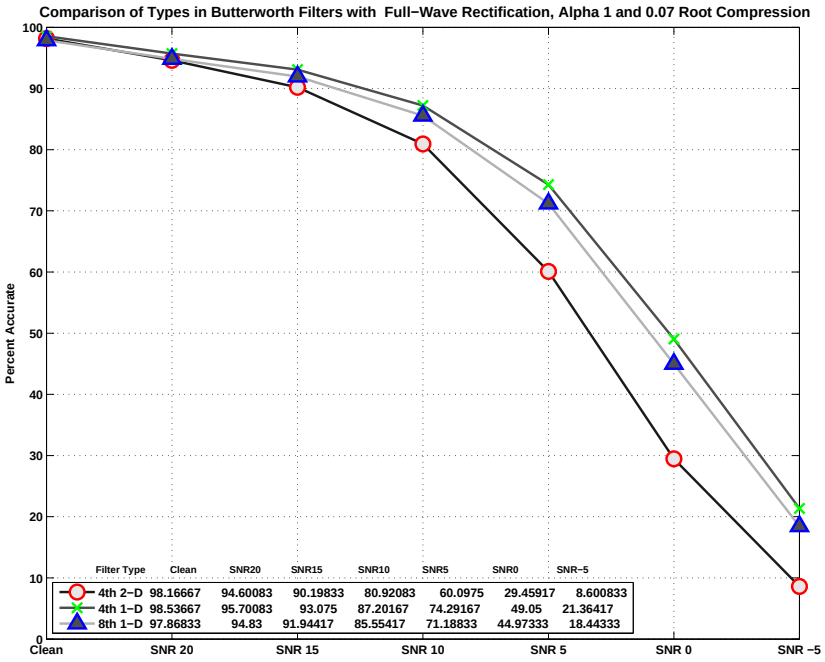

Figure 24: Comparison of Types of Butterworth Filters

from these figures that MFCCs outperform the other features for clean speech SNR20 and SNR15. Whereas for all others, bilinear transforms with root compression show superior results. The features extracted using Butterworth filters under-perform in all cases. Comparing the two figures, we observe that log compression outperforms root compression for clean conditions and that root compression works best in all other cases.

Figure 21 shows the effect of using 0.07 root compression on MFCCs. This figure shows that by using root compression instead of log compression we can improve the results for SNR10 and below; however, in cases of SNR15, SNR20 and clean conditions, there is a slight degradation in ASR results.

3.7 Types of Filtering

An important modification to MFCCs proposed by authors in [11] and [12] is the use of bandpass filter banks instead of triangular filters. In chapter we have explained in detail

the reasons behind low performance of MFCCs in low SNR conditions. In our analysis we mostly use bandpass filter banks implemented using bilinear transforms as it is done in [11] . However, in this section we focus our analysis on implementing bandpass filter banks using Butterworth filters for implementation. Our goal in this section is to investigate the bandpass filter modification effect on ASR results using three implementations of Butterworth filters. First, we use 4th order Butterworth filters using 1-D digital filter. Second, we use 8th order Butterworth filters using 1-D filter, and in the third case we use 4th order Butterworth filters using zero-phase digital filtering.

Figure 24 shows the result for above three cases for full-wave rectification, alpha 1 and 0.07 root compression. It can be seen in this figure that the results for 4th order 1-D digital filter outperforms the other filters in all conditions. This could be explained by noting that a 4th order 1-D digital Butterworth filter has a more gradual roll-off than the other two cases. However, we have shown in the previous section that these results still under-perform when compared to MFCCs and bandpass filters extracted using bilinear transforms.

CHAPTER IV

CONCLUSION

In [11] Ravindran et al. proposed a series of biologically-inspired modifications to Mel-frequency cepstral coefficients that may improve speech recognition results. In this work we have investigated the effect of each of the proposed modifications in [11] on automatic speech recognition (ASR) results and have tied the effect of modifications with input SNR levels. Using the gained insight we provide recommendations for modifications that can improve ASR results with regards to SNR levels of input.

We have used the same layout as shown in Figure 1 for extracting our features, but we have investigated each step along the way using multiple scenarios per step in order to gain insight into its effect on final recognition results. We have looked into the role of window size, spatial derivative, rectification types, smoothing, down-sampling, compression types and filter bank types. Furthermore, some modifications have shown stronger effect on final results than others. For instance, spatial derivative and down-sampling are the least effective on final results. Whereas, rectification and smoothing are moderately effective. Yet the most effective and significant modification proves to be compression type and amount.

It is concluded that window size of 10 ms yields roughly the best overall results for all cases. It is concluded that for low SNR (10 and below) the best results are achieved by using bandpass filter banks, applying spatial derivative *before* envelope detection, using full-wave rectification along with high smoothing in the envelope detector, and utilizing minimum value for down-sampling and 0.07 for root compression. To achieve best results

for high SNR input signals (15 and above), if bandpass filter banks are being used, we recommend removing spatial derivative, using half-wave rectification and less smoothing in the envelope detector, minimum value for down-sampling and 0.07 for root compression. Having said that, for high SNR signals MFCCs outperform bandpass filter bank implementations even with above recommended modifications. Hence, in cases of high SNRs we recommend using modified MFCCs, in which log compression is replaced with 0.07 root compression. For best results in clean speech, we recommend using MFCCs with log compression. Table 2 shows our final recommendations for achieving best ASR results depending on input SNR levels.

Table 2: Recommendations

Clean	MFCCs with log compression
High SNR (15 and above)	MFCCs with 0.07 root compression
Low SNR (10 and below)	Bandpass filter banks, Spatial derivative before envelope detection, Full-wave rectification, High smoothing, Minimum value for down-sampling, 0.07 root compression

REFERENCES

[1] CHEN, C., FILALI, K., and BILMES, J., "Frontend post-processing and backend model enhancement on the aurora 2.0/3.0 databases," in *International Conference on Speech and Language Processing*, pp. 241–244, 2002.

[2] CUI, X., ISELI, M., ZHU, Q., and ALWAN, A., "Evaluation of noise robust features on the aurora databases," in *in ICSLP*, 2002.

[3] GARNER, P. N., "Cepstral normalisation and the signal to noise ratio spectrum in automatic speech recognition," *Speech Communication*, vol. 53, pp. 991–1001, Oct. 2011.

[4] HECKMANN, M., DOMONT, X., JOUBLIN, F., and GOERICK, C., "A hierarchical framework for spectro-temporal feature extraction," *Speech Communication*, vol. 53, no. 5, pp. 736 – 752, 2011.

[5] HERMANSKY, H. and MORGAN, N., "RASTA processing of speech," in *IEEE Transactions on Speech and Acoustics*, vol. 2, pp. 587–589, October 1994.

[6] HUNT, M., LENNIG, M., and MERMELSTEIN, P., "Experiments in syllable-based recognition of continuous speech," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, vol. 5, pp. 880 – 883, apr 1980.

[7] KALINLI, O. and NARAYANAN, S. S., "Early auditory processing inspired features for robust automatic speech recognition," in *Proceedings of European Signal Processing Conference (EUSIPCO)*, (Poznan, Poland), Sept. 2007.

[8] LYON, R., "A computational model of filtering, detection, and compression in the cochlea," *ICASSP*, 1982.

[9] MESGARANI, M. N., MESGARANI, N., and SHAMMA, S., "Speech discrimination based on multiscale spectro–temporal," in *In Proc. ICASSP*, pp. 601–604, 2004.

[10] PEARCE, D., GNTER HIRSCH, H., and GMBH, E. E. D., "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *in ISCA ITRW ASR2000*, pp. 29–32, 2000.

[11] RAVINDRAN, S., ANDERSON, D., and SLANEY, M., "Improving the noise-robustness of mel-frequency cepstral coefficients for speech processing," *SAPA*, pp. 48–52, 2006.

[12] RAVINDRAN, S., ANDERSON, D., and SLANEY, M., "Varying time constants and gain adaptation in feature extraction for speech processing," *ICASSP*, 2007.

[13] RECOMMENDATION G.712, I., "ransmission performance characteristics of pulse code modulation channels," Nov. 1996.

[14] RUSSELL, B., FREEMAN, W., A.A. EFROS, SIVIC, J., and ZISSERMAN, A., "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, pp. 1605–1614.

[15] SHARMA, J., DRAGOI, V., TENENBAUM, J. B., MILLER, E. K., and SUR, M., "V1 neurons signal acquisition of an internal representation of stimulus location.," *Science (New York, N.Y.)*, vol. 300, pp. 1758–63, June 2003.

[16] V1.1.3, E. E. . ., "Speech processing, transmission and quality aspects (stq); distributed speech recognition; front-end feature extraction algorithm; compression algorithms," *Speech Communication*, vol. 9, 2003.

[17] WANG, K. and SHAMMA, S. A., "Self Normalization and noise-robustness in early auditory representation," *sap*, vol. 2, pp. 421–435, 1994.

[18] YANG, X., WANG, K., and SHAMMA, S., "Auditory representations of acoustic signals," *Information Theory, IEEE Transactions on*, vol. 38, pp. 824 –839, mar 1992.

[19] YIN, H., HOHMANN, V., and NADEU, C., "Acoustic features for speech recognition based on gammatone filterbank and instantaneous frequency," *Speech Commun.*, vol. 53, pp. 707–715, May 2011.

[20] YOUNG, S., "Hidden markov model toolkit (htk)," Available: http://htk.eng.cam.ac.uk/.